Predicting Variants of Alternative Splicing from NGS data without the Genome

Stefano Beretta* Paola Bonizzoni* Gianluca Della Vedova†
Raffaella Rizzi*

August 2, 2011

Abstract

Known approaches to determine the gene structure, due to alternative splicing (AS), rely on some forms of spliced alignment of the transcripts against the genomic sequence of the gene producing the transcripts. Anyway, the abundance of data arising from next-generation sequencing technologies allows a new approach, proposed in this paper, that does not require any kind of alignment of the transcripts against the genome. We formalize the underlying computational problem, proving the soundness of our approach from both a theoretical and experimental point of view.

1 Introduction

Next Generation Sequencing (NGS) technologies allow massive and parallel sequencing of biological molecules (DNA, RNA and proteins), and they have a strong impact on molecular biology and bioinformatics [8,9]. In particular, RNA-Seq is a recent technique to sequence expressed transcripts, characterizing both the type and the quantity of transcripts expressed in a cell (its transcriptome). The main goal in transcriptomics is to predict the exon-intron structure of a gene and its full-length transcripts (or isoforms), and some of the challenging tasks in RNA-Seq data analysis are mapping the sequenced short reads to a reference genome, inferring full-length isoforms and its expression level [2,4,10], and assembling the reads into contigs

*DISCo, Univ. Milano-Bicocca, Milan, Italy email{beretta,bonizzoni,rizzi}@disco.unimib.it
†Dip. Statistica, Univ. Milano-Bicocca, Milan, Italy emailgianluca.dellavedova@unimib.it

before aligning them to the genome and to predict the the gene structure and the transcript isoforms [14].

Some of the current methods, used to identify the exon-intron boundaries on the genome, are QPALMA [3], TopHat [12], Supersplat [1] and MapSplice [13]. They all receive a genomic sequence and a set of RNA-Seq data and produce in output the spliced alignments of the input reads against the genome. Few efforts have been devoted to assemble RNA-Seq data into long contigs to be aligned to a reference genome, or even to use RNA-Seq for obtaining a "raw" structure of a gene. In this paper, we address the last problem, that is predicting from NGS data the gene structure induced by the different full-length isoforms due to alternative splicing (AS) [7]. More precisely, we analyze RNA-Seq data that have been sampled from the transcripts of a gene, with the goal of building a graph representation of the variants of alternative splicing corresponding to those full-length isoforms. The novelty of our method relies on the fact that we build such a graph in absence of the genome. A subsequent step is to efficiently map the graph to the genome in order to refine the gene structure and to compute intron data that, obviously, cannot be present in the isoforms. As we have already stated, our approach does not use the genome. In particular, no alignment of the RNA-Seq reads against the genome is required, potentially improving the overall running time as well as skipping a post-processing phase that filters erroneous or redundant data that is an undesirable side product of aligning huge amounts of reads. Last, but not least, an advantage of our approach is that it is applicable also to data originating from an unknown genome. We show the effectiveness of our approach from two different points of view: in fact we will identify some conditions which, if satisfied, guarantee that our algorithm produces the correct gene structure. Moreover we will investigate experimentally the behaviour of our algorithm on simulated NGS data, focusing on the scalability of our approach to the huge quantity of data that characterizes NGS technologies.

2 The Isoform Graph and the IGR Problem

In this section we give a formal notion of a gene $\mathcal{G}$ and we introduce the Isoform Graph for representing $\mathcal{G}$. Moreover, we formalize the problem of reconstructing the Isoform Graph from the RNA-Seq data derived from the transcripts of $\mathcal{G}$.

Let $s = s_1 s_2 \cdots s_{|s|}$ be a sequence of characters, that is a *string*. Then $s[i : j]$ denotes the substring $s_i s_{i+1} \cdots s_j$ of s, while $s[: i]$ and $s[j :]$ denote

respectively the *prefix* of s consisting of i symbols and the *suffix* of s starting with the j-th symbol of s. We denote with $\mathrm{pre}(s,i)$ and $\mathrm{suf}(s,i)$ respectively the prefix and the suffix of length i of s. Among all prefixes and suffixes, we are especially interested into $\mathrm{LH}(s) = \mathrm{pre}(s, |s|/2)$ and $\mathrm{RH}(s) = \mathrm{suf}(s, |s|/2)$ which are called the *left half* and the *right half* of s. Given two strings s_1 and s_2, the *overlap* $ov(s_1, s_2)$ is the length of the longest suffix of s_1 that is also a prefix of s_2. The *fusion* of s_1 and s_2, denoted by $\varphi(s_1, s_2)$, is the string $s_1[: |s_1| - ov(s_1, s_2)]s_2$ obtained by concatenating s_1 and s_2 after removing from s_1 its longest suffix that is also a prefix of s_2. We extend the notion of fusion to a sequence of strings $\langle s_1, \ldots, s_k\rangle$ as $\varphi(\langle s_1, \ldots, s_k\rangle) = \varphi(s_1, \varphi(\langle s_2, \ldots, s_k\rangle))$ if $k > 2$ and $\varphi(\langle s_1, s_2\rangle) = \varphi(s_1, s_2)$.

In this paper a gene consists of a discrete genomic region and the information for the synthesis of functional proteins, mainly its full-length isoforms or transcript products of the gene [11]. Let us recall that the genomic region of a gene consists of a sequence of coding regions alternating with non-coding ones. An isoform of the gene is a concatenation of some of the coding regions of the gene respecting their order in the genomic region. Alternative splicing regulates how different coding regions are included to produce different full-length isoforms or transcripts which are modeled here as sequences of *blocks*.

Definition 2.1. A *block* consists of a string, typically taken over the alphabet $\Sigma = \{a, c, g, t\}$, and an integer called position of the block.

Clearly different blocks might have the same nucleotide sequence (they might represent a repeated sequence on the genome). Given a block b, we denote by $s(b)$ and $p(b)$ its string and position respectively. Notice that our formal model of a gene is based on the set of isoforms and only loosely consider the genome. Indeed, we do not take into account the real position of blocks on the genome, but we implicitly represent all coding regions of a gene as a concatenated sequence of blocks, and each isoform consists of a subsequence of blocks.

In our framework a *gene coding region* is a sequence (that is, an ordered set) $B = \langle b_1, b_2, \cdots, b_n\rangle$ of blocks with $p(b_1) = 0$ and $p(b_i) = p(b_{i-1}) + |s(b_{i-1})|$ for each $i > 1$. Then, the *string coding region for* B is the string $s(b_1)s(b_2)\cdots s(b_n)$ obtained by orderly concatenating the strings of the blocks in B. Intuitively a gene coding region is the sequence of all coding regions on the whole genomic sequence for the studied gene (see Figure 1). We define an *isoform* f *compatible* with B, as a subsequence of B, that is $f = \langle b_{i_1}, \cdots, b_{i_k}\rangle$ where $i_j < i_{j+1}$ for $1 < j \leq k$. By a slight abuse of language we define the string of f, denoted by $s(f)$, as the concatenation of

the strings of the blocks of f. Observe that the position of a block is used in our model to encode the order of the blocks forming a coding region induced by a set of isoforms.

Definition 2.2. An *expressed gene* is a pair $\langle B, F\rangle$ where B is a gene coding region and F is a set of isoforms compatible with B where each block of B appears in some isoforms of F and for each pair (b_j, b_{j+1}) of consecutive blocks of B, there exists an isoform $f \in F$ s.t. exactly one of b_j and b_{j+1} appears in f.

The rationale for the two conditions imposed over expressed gene is that we want to characterize the information contained within a gene structure and that is induced by the blocks of isoforms (without knowledge of the genomic sequence from which the isoforms are extracted). Clearly, it is impossible to identify a block that is not in any isoform or to distinguish two blocks that are consecutive in the genomic sequence and are not separated in any isoform because, in both cases, such information is not contained in the isoform. Moreover, Definition 2.2 implies that the set B of blocks of a string coding region of an expressed gene $\langle B, F\rangle$ is unique and is a minimum-cardinality set explaining all isoforms in F. Thus, the pair $\langle B, F\rangle$ describes a specific gene.

Given an expressed gene $\mathcal{G} = \langle B, F\rangle$, we define the *Isoform Graph* of $\mathcal{G}$ as a directed graph $G_F = \langle B, E\rangle$, where B is seen as a set without considering its order, and a pair (b_i, b_j) is an arc of E, iff b_i and b_j are consecutive in at least an isoform of F. Notice that G_F is a directed acyclic graph, since the sequence B is also a topological sort of G_F (see Figure 2 for a simple example).

Informally an edge (b, b') is an evidence of a junction between blocks b and b' in at least one isoform of the gene, as isoforms correspond to paths in G_F. A path p of G_F is called *segment* if all vertices of p have indegree and outdegree equal to 1. Clearly no vertex can belong to more than one segment. Therefore we can define the *Reduced Isoform Graph* G_F^* associated to G_F as the result of contracting each segment of G_F into a single vertex.

To assess the soundness of our definitions, we show how some interesting biological phenomena corresponds to specific configurations in the Isoform Graph. In fact, there exists a skipping of a block b_1 in some isoform, if there exist two other blocks b_2 and b_3 s.t. (b_2, b_1), (b_1, b_3) and (b_2, b_3) are all arcs of G_F. Since in our framework blocks might also be portions of exons, competing events and skipping of consecutive exons at the genome level might be represented as a skipping of one block at the isoform level (Fig. 2(a) and 2(b)). Finally, a mutual exclusion of two blocks b_1 and b_2

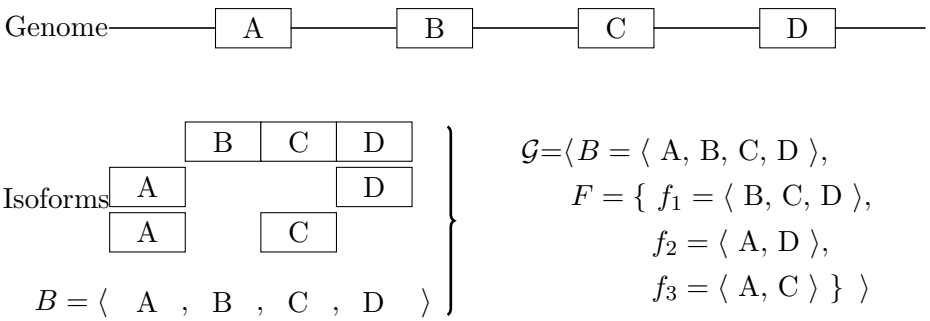


Figure 1: Example of expressed gene. Blocks are the exons A, B, C and D.

between two isoforms corresponds to a subgraph of G_F consisting of four blocks b_1, b_2, b_3 and b_4 s.t. (b_3, b_1), (b_3, b_2), (b_1, b_4) and (b_2, b_4) are all arcs of $\mathcal{G}$ (Fig. 2(b)).

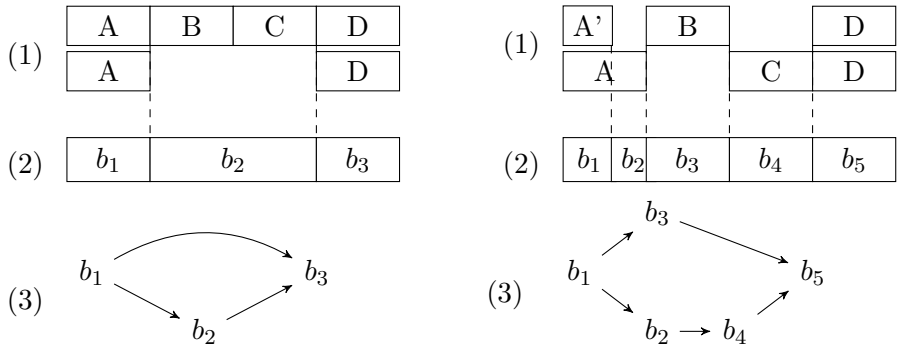
The main idea of our paper is that we can reconstruct the Isoform Graph (or an approximation of that) of a gene $\mathcal{G}$ from a sufficiently large set of RNA-Seq data obtained from the gene transcripts. Let $\langle B, F\rangle$ be an unknown expressed gene. Then, a *RNA-Seq read* (simply called *read*), extracted from $\langle B, F\rangle$, is a substring of the nucleotide sequence $s(f)$ of some isoform $f \in F$. We can now formally introduce our computational problem.

problem 1. The Isoform Graph Reconstruction (IGR) Problem
Input: a set R of reads extracted from an unknown expressed gene $\langle B, F\rangle$.
Output: the Isoform Graph G_F of $\langle B, F\rangle$.

3 Methods

In this section we present a method for solving the IGR problem. Our approach to compute the graph G_F consists of first identifying the vertex set B_F and then the edge set E_F of G_F.

For ease of exposition, the discussion of the method assumes that reads have no errors, all have length l and the isoforms contain no repeated sequences of length l. Our actual implementation contains some steps that complement our procedure to deal with reads that violate such requirements. In particular we have a preprocessing phase to remove from R the most common type of errors. The basic idea of our method is that we can find two disjoint subsets R_1 and R_2 of R where reads of R_1, called *unspliced*, can be assembled to compute the nodes in B_F, while reads of R_2, called *perfectly*



(a) Exon Skipping.

(b) Competing and Mutually Exclusive Exons.

Figure 2: Examples of Isoform Graphs. In (a) it is shown a skipping of the two consecutive exons B and C of the second isoform w.r.t. the first one (1). The resulting sequence of blocks is $\langle b_1, b_2, b_3 \rangle$ (2). The Isoform Graph is reported in (3). In (b) it is reported a competing variant between exons A and A' and two mutually exclusive exons B and C (1). The sequence of blocks is $\langle b_1, b_2, b_3, b_4, b_5 \rangle$ (2), and the Isoform Graph is in (3).

spliced or *spliced*, are an evidence of a junction between two blocks (that is an arc of G_F).

Definition 3.1. Let r be a read of R. Then r is *spliced* if there exists another $r' \in R$, $s(r) \neq s(r')$, such that $\text{pre}(r, k) = \text{pre}(r', k)$ or $\text{suf}(r, k) = \text{suf}(r', k)$, for $l/2 \leq k$. Moreover a read r is *perfectly spliced* if there exists another $r' \in R$, $s(r) \neq s(r')$, such that the longest common prefix (or suffix) of r and r' is exactly of length $l/2$.

In our framework, a junction site between two blocks b_1 and b_3 that appear consecutively within an isoform is detected when we find a third block b_2 such that, in some isoform, b_2 appears immediately after b_1 or immediately before b_3. For illustrative purposes, let us consider the case when b_2 appears immediately after b_1 in an isoform (Figure 2(a)). The best case scenario consists of two reads r_1 and r_2 such that $ov(b_1, r_1) = ov(r_1, b_3) = l/2$ and $ov(b_2, r_2) = ov(r_2, b_3) = l/2$, that is r_1 is cut into halves by the junction site separating b_1 and b_3, while r_2 is cut into halves by the junction site separating b_2 and b_3. Notice that in such case r_1 and r_2 are both perfectly spliced. In a less-than-ideal scenario, we still can find two reads r_1 and r_2 sharing a common prefix (or suffix) that is longer than $l/2$, in which case

the two reads are spliced and called *left-spliced* (respect. *right-spliced*). A read $r \in R$ is called *unspliced* if $\mathrm{LH}(r) \neq \mathrm{LH}(r_1)$ and $\mathrm{RH}(r) \neq \mathrm{RH}(r_1)$ for all other reads $r_1 \in R$. We can now give the definition of *maximal chain* of unspliced reads.

Definition 3.2 (Chain)**.** A sequence $C = \langle r_1, r_2, \cdots, r_n \rangle$ of unspliced reads is a *chain* if $ov(r_i, r_{i+1}) \geq l/2$ for $1 \leq i < n$. Moreover C is *maximal* if no supersequence of C is also a chain.

We define the string of a chain C the string $s(C) = \varphi(C)$. Given a maximal chain C, its string is the longest substring of an hypothetical block string, that can be computed using the set R. Our intuition is that maximal chains are candidates of the string of hypothetical blocks, each perfectly spliced read witnesses that two hypothetical blocks are likely to appear consecutively in some isoforms. The notion of link between two chains formalizes such idea.

Definition 3.3. Let C, C' be two (maximal) chains and let r be a perfectly spliced read such that $\mathrm{LH}(r) = \mathrm{suf}(s(C), l/2)$ and $\mathrm{RH}(r) = \mathrm{pre}(s(C'), l/2)$. Then r is called a *link* for the pair (C, C'). Moreover, we also say that C and C' are respectively *left-linked* and *right-linked* by r.

At this point we can define the *RNA-Seq graph* which corresponds to the structure actually computed by the algorithm that we will describe next.

Definition 3.4. Let $\langle B, F \rangle$ be an expressed gene, and let R be a set of RNA-Seq reads) extracted from F. Then the *RNA-Seq Graph* of R is a directed graph $G_R = (\mathcal{C}, E_R)$, where $\mathcal{C} = \{C_1, \cdots, C_n\}$ is a set of maximal chains that can be derived from R, and $(C_i, C_j) \in E_R$ iff there exists in R a link for (C_i, C_j).

Now we want to show how to construct a RNA-Seq Graph G_R in polynomial time, as well as proving that G_R is a good approximation of the Isoform Graph G_F. The algorithm is organized into three steps that are detailed below. In the first step we build a data structure to store the reads in R. We use two hash tables which guarantee a fast access to the input reads. The second step creates the nodes of G_R by composing the maximal chains of the unspliced reads of R. The last step creates the vertices of G_R and consists in linking the maximal chains obtained in the second step.

3.0.1 Step 1: Fingerprinting of RNA-Seq reads

Given an l-long read r, its fingerprint is a pair $\phi(r) = (\phi_1(r), \phi_2(r))$ of binary integers. Since all our strings are over the alphabet $\{a, c, g, t\}$, we

can encode each character with a 2-bit binary integer as follows: $\text{enc}(a) = 0 = 00_2$, $\text{enc}(c) = 1 = 01_2$, $\text{enc}(g) = 2 = 10_2$, $\text{enc}(t) = 3 = 11_2$. Then we can encode a string s as $\text{enc}(s) = \sum_{i=1}^{|s|} 2^{i-1}\,\text{enc}(s[i])$. Given a read r, we define $\phi_1(r) = \text{enc}(\text{LH}(r))$ and $\phi_2(r) = \text{enc}(\text{RH}(r))$, that are respectively the encoding of the first half (also called *left fingerprint*) and of the second half (also called *right fingerprint*) of r. The described encoding is a one-to-one mapping between strings s and numbers between 0 and $2^{|s|} - 1$. Therefore, we will use interchangeably a string and its fingerprint.

Given a set R of RNA-Seq data and given $\phi(r) = (\phi_1(r), \phi_2(r))$ for each $r \in R$, we use two data structures for storing the fingerprints of all the reads in R. More precisely, let $\phi_1(R)$ and $\phi_2(R)$ be the sets of the left and the right fingerprints, respectively, for all the reads in R. We employ two hash tables T_l and T_r, indexed respectively by fingerprints in $\phi_1(R)$ and in $\phi_2(R)$. We denote with $\mathcal{L}_l(\phi)$ (resp. $\mathcal{L}_r(\phi)$) the set of reads in R whose left (resp. right) fingerprint is ϕ. Notice that $\mathcal{L}_l(\phi)$ (resp. $\mathcal{L}_r(\phi)$) is the set of reads sharing a prefix (resp. a suffix) of length at least $l/2$. Moreover, a read r is spliced iff $|\mathcal{L}_l(\phi_1(r))| > 1$ (or $|\mathcal{L}_r(\phi_2(r))| > 1$), and is *perfectly spliced* if there exists a read $r_1 \in \mathcal{L}_l(\phi_1(r)))$ (or $r_1 \in \mathcal{L}_r(\phi_2(r))))$ such that $r[l/2+1] \neq r_1[l/2+1]$ (or $r[l/2-1] \neq r_1[l/2-1]$), that is, the longest common prefix (resp. suffix)) of r and r_1 is exactly $l/2$ characters long. A read r is called *unspliced* iff $|\mathcal{L}_l(\phi_1(r))| = |\mathcal{L}_r(\phi_2(r))| = 1$.

We can build the tables T_l and T_r in $O(|R|l)$ time (that is a linear time w.r.t. the space necessary to store the reads). In fact we have to scan R only once and, for each read r, we compute its left and right fingerprints, which requires $O(l)$ time. Successively we find the corresponding entry in T_l and T_r, and we add a read in the set associated to each entry. Notice that each such set contains at most $2^{l/2}$ elements, therefore storing those sets with a search tree leads to a $O(\log(2^{l/2})) = O(l)$ insertion time. In practice the value of l is fixed to 64 (i.e. the CPU word size), therefore the time to construct T_l and T_r is $O(|R|)$.

3.0.2 Step 2: Building the set $\mathcal{C}$ of Maximal Chains

The procedure *BuildChains* described in Algorithm 1 takes as input a set R of RNA-Seq reads and produces the set $\mathcal{C}$ of all the maximal chains that can be obtained from R.

Let $R_1 \subseteq R$ be the set of the *unspliced* reads. The algorithm selects any read r of R_1. For efficiency purposes we assume that r is the first read of R_1, so that its selection takes constant time. Initially the algorithm starts with a chain C containing only r. Let r_r be a read in R_1 maximizing $ov(r, r_r)$. If

$ov(r, r_r) \geq l/2$ then r_r is called a *right extension* of r; in such case $\langle r, r_r \rangle$ is a chain. Notice that the right extension (if it exists) is unique since we can choose only unspliced reads. By maximality of $ov(r, r_r)$, there does not exist an *unspliced* read r^* s.t. $\langle r, r^*, r_r \rangle$ is a chain. Just as we have done for right extensions, we can define the notion of *left extension*. It is an immediate consequence of our arguments that iteratively adding the right extension of the rightmost read in C, as well as iteratively adding the left extension of the leftmost read in C results in a maximal chain.

The time required by this procedure is $O(|R_1|l)$. In fact, each read in R_1 is considered only once, and finding the left or right extension of a read r can be performed in $O(l)$ time. Let us consider the case of finding a right extension of r (the case of left extension is symmetrical). For decreasing values of k ranging from $l-1$ to $l/2$ we determine if there exists a read r^* such that $ov(r, r^*) = k$ in $O(1)$ time. At the end of step 2, we check if some blocks at the beginning or the end of the chain must be removed in order to guarantee that the string of each block does not span more than one block. The description of this phase is omitted due to space constraints.

3.0.3 Step 3: Linking Maximal Chains

Algorithm 2 computes the set E_R of arcs of G_R, given the set R_2 of the *perfectly spliced* reads and the set $\mathcal{C}$ of the maximal chains (i.e. the vertices of G_R) obtained by the set R_1 of the *unspliced* reads. More precisely, given a perfectly spliced read r, we denote with $\mathcal{D}(r)$ and $\mathcal{A}(r)$ the set of maximal chains (i.e. vertices of G_R) that are, respectively, left-linked and right-linked by r according to Definition 3.3. In other words, r is a *link* for each pair of chains in $\mathcal{D}(r) \times \mathcal{A}(r)$. Moreover each such pair will be an arc of G_R. Algorithm 2 is greatly sped up if Algorithm 1 also store the encoding of the prefix and the suffix of length $l/2$ of each maximal chain in $\mathcal{C}$. The time required is $O(|\mathcal{C}| + |R| + |E_R|)$.

4 Validation

In this section we investigate some properties of the instances of IGR, as well as of the graphs G_R and G_F, with the final goal of proving how they are related. A first basic property states that graph G_R cannot have segments, which have been defined as paths consisting of nodes with out-degree and in-degree one.

Indeed, assume to the contrary that b_i, b_{i+1} are two consecutive nodes of a segment of G_R, that is both b_i and b_{i+1} have indegree and outdegree 1

Algorithm 1: BuildChains(R)

```
Data: R, a set of RNA-Seq reads
R_1 ← {r ∈ R | r is unspliced};
C ← ∅;
while R_1 ≠ ∅ do
    r ← any read from R_1;
    R_1 ← R_1 \ {r};
    C ← ⟨r⟩;
    r_1 ← r;
    // Try to extend the chain on the right
    while ∃ a right extension r_2 ∈ R_1 of r do
        append r_2 to C;
        R_1 ← R_1 \ {r_2};
        r_1 ← r_2;
    // Try to extend the chain on the left
    while ∃ a left extension r_2 ∈ R_1 of r do
        prepend r_2 to C;
        R_1 ← R_1 \ {r_2};
        r ← r_2;
    C ← C ∪ C;
return C;
```

and (b_i, b_{i+1}) is an arc of G_R. By Definitions 3.3, 3.4 and by construction of our procedure, since (b_i, b_{i+1}) is an arc of graph G_R, then there exists a perfectly spliced read $r \in R$ with overlap $l/2$ with the strings of the two nodes b_i and b_{i+1}. Moreover, by definition of perfectly spliced read, there must exists another perfectly spliced read r' with $LH(r) = LH(r')$ or $RH(r) = RH(r')$. But, the first case implies that b_i has out-degree at least two, while the second case implies that b_{i+1} must be of in-degree at least two, contradicting the initial assumption that b_i, b_{i+1} are two consecutive nodes of a segment. Since G_R cannot have segments, we contract each segment of G_F into a single vertex, thereby obtaining the Reduced Isoform graph G_F^*. In the following of this section we will compare G_R with G_F^* under some hypothesis on the instance.

Let R be an instance of IGR originating from an expressed gene $\langle B, F\rangle$. Then R is a *good* instance if: (i) all blocks in the gene coding region B of $\langle B, F\rangle$ are at least l characters long, (ii) for each three blocks b, b_1 and b_2 s.t. b and b_1 are consecutive in an isoform, b and b_2 are consecutive in

Algorithm 2: LinkChains(R_2, $\mathcal{C}$)

Data: R_2, the set of perfectly spliced reads, and $\mathcal{C}$, the set of all the maximal chains

$E_R \leftarrow \varnothing$;
foreach $r \in R_2$ **do**
  $\mathcal{D}(r) \leftarrow \varnothing$;
  $\mathcal{A}(r) \leftarrow \varnothing$;
foreach $C \in \mathcal{C}$ **do**
  $f \leftarrow \text{enc}(\text{pre}(s(C), l/2))$;
  foreach $r \in R_2$ *such that* $\text{enc}(\text{LH}(r)) = f$ **do**
    $\mathcal{D}(r) \leftarrow C$;
  $f \leftarrow \text{enc}(\text{suf}(s(C), l/2))$;
  foreach $r \in R_2$ *such that* $\text{enc}(\text{RH}(r)) = f$ **do**
    $\mathcal{A}(r) \leftarrow C$;
foreach $r \in R_2$ **do**
  if $\mathcal{D}(r) \neq \varnothing$ *and* $\mathcal{A}(r) \neq \varnothing$ **then**
    foreach $p \in \mathcal{D}(r) \times \mathcal{A}(r)$ **do**
      $E_R \leftarrow E_R \cup p$;
return E_R

another isoform, then b_1 and b_2 begin with different characters. Also, for each three blocks b, b_1 and b_2 s.t. b_1 and b are consecutive in an isoform, b_2 and b are consecutive in another isoform, then b_1 and b_2 end with different characters; (iii) for each subsequence B_1 of B, the string $s(B_1)$ does not contain two identical substrings of length $l/2$; (iv) all isoforms in F start with the same block b_s and end with the same block b_e (v) all the l-long substrings of some isoforms are also reads in R.

We can prove that graph G_R is isomorphic to G^*_F when R is a good instance. Due to space constraints, we omit the proof. Clearly, real data does not satisfy all of the conditions of a good instance. Mainly, there are two conditions that are violated more frequently, leading to a graph G_R that is slightly different from G^*_F: one is condition (i) about the length of the blocks and the second is condition (iii) about the presence of repeated substrings of length at least $l/2$. Consequently the graph G_R is an approximation of graph G^*_F.

5 Experimental Results

We implemented our method as a C++ program that has been run on a workstation with two 2.8GHz quad-core processors and 12GB of RAM. Our tool takes as input a set of RNA-Seq reads, and outputs the RNA-Seq graph G_R. We assessed the accuracy and the efficiency of our approach on simulated RNA-Seq data obtained from the isoforms annotated for the set of 112 genes – extracted from the 13 ENCODE regions – used as training set in the EGASP competition [5]. For each gene, we reconstructed the actual isoform graph G_F from the set of the full-length isoforms annotated on the reference genome. We have subsequently obtained from G_F the graph G_F^* as described in Section 3.

On simulated data, we have performed two experiments, the first without errors and the second allowing the presence of errors in the read. The goal of the two experiments are different; when analyzing error-free reads we are assessing the quality of our *model*, while the second experiment allow us to understand the quality of our *implementation* on real data.

In the first experiment, for each gene in input, we extracted a set R of RNA-Seq reads consisting of all possible substrings of length 64 of the isoforms produced by the gene. Then each set of reads is given to our program to compute the RNA-Seq graph G_R. Successively we compared each graph G_R with the graph G_F^* obtained from the annotated isoforms.

The comparison of the graphs G_R and G_F^* is not straightforward, therefore we discuss next the procedure we have designed. For clarity's sake, we consider that vertices of both graphs are strings instead of blocks. Let s, t be two strings. Then s and t are p-trim equivalent if it is possible to obtain the same string by removing from s and t a prefix and/or a suffix no longer than p (notice that the removed prefixes and suffixes might be empty and can differ, in length and symbols, between s and t). Let v and w be respectively a vertex of G_R and of G_F^*. Then v *maps* to w, if v and w are 5-trim equivalent. Moreover we say that v *predicts* w if v maps to w and no other vertex of G_R maps to w. We can generalize those notions to arcs and graphs, where an arc $(v_1.v_2)$ of G_R maps (resp. predicts) an arc (w_1, w_2) of G_F^* if v_1 maps to (resp. predicts) w_1 and v_2 maps to (resp. predicts) w_2. Finally, the graph G_R correctly predicts G_F^* if those two graphs have the same number of vertices and arcs and each vertex/arc of G_R predicts a vertex/arc of G_F^*.

The accuracy of our method is evaluated by two standard measures, *Sensitivity* (Sn) and *Positive Predictive Value* (PPV) considered at vertex and arc level. Sensitivity is defined as the proportion of vertices (or arcs)

of G_F^* that have been correctly predicted by a vertex (or arc) in G_R, while positive predictive value is the proportion of the vertices (or arcs) of G_R that correctly predict a vertex (or an arc) in G_F^*. Table 1 in the Appendix summarizes the results of the first experiment. In particular, 43 out of 112 genes have Sn and PPV of 1.0 both at vertex and arc level, that is G_R correctly predicts G_F^*. That results means that the set of reads extracted from the other 69 gene were not good instances, mostly due to the presence of short blocks or relatively long repeated regions. Nonetheless, the Sn and PPV values suggest that the predicted gene structure G_R is similar to the actual gene structure G_F^*, as witnessed by the average (over the whole dataset) Sn and PPV values that are 0.880 and 0.927 at vertex level, respectively and 0.786 and 0.868 at arc level, respectively. Also, the median values are 0.914, 1, 0.857, and 0.977 respectively.

To better give an intuitive idea of the similarity of our prediction and the actual structure, we have selected the gene $L1CAM$ as a representative gene among those where our implementation does not correctly predict the graph G_F^*. To this purpose, Fig. 3 in the Appendix shows G_R and G_F^* for the gene L1CAM. The only differences between the graphs are due to two small blocks (of length 1 and 15) that our method cannot predict. It is clear that the overall structure has been satisfactorily predicted.

The second experiment was performed on the same set of genes, but for different values of two parameters, c and p, as follows. For each gene, we started from the set of reads obtained in the first experiments, and we replicated c times all those reads, obtaining a multiset R^c, to simulate an increased, and closer to actual values, coverage. Then we mutated $p\%$ of the reads in R^c, changing one nucleotide. The mutated read, the position of the mutation and the new nucleotide have been selected uniformly at random. The final result is the set $R^{c,p}$. We considered all possible pairs of values of our parameters, with $c = 4, 5, 6$ and $p = 2, 4, 8, 16$ (12 data sets). Overall our program has processed about 23 million reads for a total running time of 67 minutes. For this experiment, the running time includes also a step necessary to compute a set of consensus read. This preprocessing phase allows to correct most of the errors introduced.

Due to space constraints we are unable to represent all results of the 12 datasets. Since the quality of the results is better for larger values of coverage and lower error rates, we consider here only the data set with $c = 4$ and $p = 16\%$ (i.e. the worst data set). Table 2 in the Appendix details those results. We point out that 40 out of 112 genes have Sn and PPV of 1.0 both at vertex and arc level, therefore the graph G_F^* has been correctly predicted. On this input set, the average Sn and PPV values are: 0.868 and 0.906 at

vertex level respectively, 0.765 and 0.836 at arc level respectively. Also, the median values are 0.9166, 0.954, 0.842, and 0.913 respectively.

6 Conclusions

In this paper we propose a new graph-model for AS variants that does not rely on the genomic sequence. We have described a new efficient algorithm that is particularly suited for elaborating RNA-Seq data to compute such graph. We have proved that, under some assumptions, our algorithm correctly computes the graph. Finally we have implemented our algorithm and performed an extensive experimental analysis on simulated data that has confirmed the soundness of our model and the quality of our implementation of real data.

We plan to attack the main issues of our approach that are mainly correlated to the presence of short exons and to relatively long repeated regions in the isoforms. Indeed, there is a set of reads (those the are neither unspliced nor perfectly spliced) that are not used by our implementation and that are crucial in improving the quality of our predictions.

Moreover we are planning to design an efficient algorithm to analyze (without resorting to an alignment) the graph computed and the genomic sequence in order to refine the graph.

References

[1] Bryant, D.W., Shen, R., Priest, H.D., Wong, W.K., Mockler, T.C.: Supersplat—spliced RNA-Seq alignment. Bioinformatics 26(12), 1500–1505 (2010)

[2] Trapnell, C., Williams, B.A., Pertea, G., Mortazavi, A., Kwan, G., van Baren, M.J., Salzberg, S.L., Wold, B.J., Pachter, L.: Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. Nature Biotechnology 28(5), 516–520 (May 2010)

[3] De Bona, F., Ossowski, S., Schneeberger, K., Ratsch, G.: Optimal spliced alignments of short sequence reads. BMC Bioinformatics 9(Suppl 10), O7 (2008)

[4] Feng, J., Li, W., Jiang, T.: Inference of isoforms from short sequence reads. Journal of Computational Biology 18(3), 305–321 (2011)

[5] Guigó, R., Flicek, P., Abril, J., Reymond, A., Lagarde, J., Denoeud, F., Antonarakis, S., Ashburner, M., Bajic, V.B., Birney, E., Castelo, R., Eyras, E., Ucla, C., Gingeras, T.R., Harrow, J., Hubbard, T., Lewis, S.E., Reese, M.G.: EGASP: the human ENCODE Genome Annotation Assessment Project. Genome biology 7(Suppl 1), S2.1–31 (Jan 2006)

[6] Haas, B.J., Zody, M.C.: Advancing RNA-Seq analysis. Nat. Biotech. 28(5), 421–423 (2010)

[7] Leipzig, J., Pevzner, P., Heber, S.: The Alternative Splicing Gallery (ASG): bridging the gap between genome and transcriptome. Nucleic Acid Research 32(13), 3977–3983 (2004)

[8] Mardis, E.R.: The impact of next-generation sequencing technology on genetics. Trends in genetics : TIG 24(3), 133–41 (Mar 2008)

[9] Metzker, M.L.: Sequencing technologies - the next generation. Nature reviews. Genetics 11(1), 31–46 (2010)

[10] Nicolae, M., Mangul, S., Mandoiu, I.I., Zelikovsky, A.: Estimation of alternative splicing isoform frequencies from RNA-Seq data. Alg. Mol. Biol. 6(1), 9 (2011)

[11] Pesole, G.: What is a gene? an updated operational definition. Gene 417(1-2), 1–4 (2008)

[12] Trapnell, C., Pachter, L., Salzberg, S.L.: TopHat: discovering splice junctions with RNA-Seq. Bioinformatics 25(9), 1105–1111 (2009)

[13] Wang, K., Singh, D., Zeng, Z., Coleman, S.J., Huang, Y., Savich, G.L., He, X., Mieczkowski, P., Grimm, S.A., Perou, C.M., MacLeod, J.N., Chiang, D.Y., Prins, J.F., Liu, J.: MapSplice: Accurate mapping of RNA-Seq reads for splice junction discovery. Nucleic Acid Research 38(18), e178 (2010)

[14] Wang, Z., Gerstein, M.B., Snyder, M.: RNA-Seq: a revolutionary tool for transcriptomics. Nature reviews. Genetics 10(1), 57–63 (Jan 2009)

A Appendix

Table 1: Details of the first experiment with no errors in the input set. Columns "G_R vertices" and "G_F vertices" are the number of vertices in the graph G_R and G_F, respectively. Column "Correctly predicted vertices" represents the number of vertices of G_F that are correctly predicted by G_R. The next 3 columns ("G_R arcs", "G_F arcs" and "Correctly predicted arcs") represent the number of arcs in the two graphs and the number of arcs of G_F that are correctly predicted by G_R. The last 4 columns are the values of Sn and PPV on vertices and arcs.

Gene	G_R vertices	G_F vertices	Correctly predicted vertices	G_R arcs	G_F arcs	Correctly predicted arcs	Sn vertices	PPV vertices	Sn arcs	PPV arcs
ARHGAP4	32	34	31	45	47	41	0.912	0.969	0.872	0.911
ATP11A	20	20	17	21	23	14	0.850	0.850	0.609	0.667
ATP6AP1	9	9	8	11	12	10	0.889	0.889	0.833	0.909
AVPR2	7	7	7	8	8	8	1.000	1.000	1.000	1.000
BPIL2	5	5	5	5	5	5	1.000	1.000	1.000	1.000
BRCC3	9	11	9	11	15	10	0.818	1.000	0.667	0.909
C20orf173	5	5	5	6	6	6	1.000	1.000	1.000	1.000
C22orf24	5	5	5	5	5	5	1.000	1.000	1.000	1.000
C22orf28	11	11	11	12	13	12	1.000	1.000	0.923	1.000
C22orf30	12	12	10	15	15	11	0.833	0.833	0.733	0.733
C6orf150	7	7	7	7	7	7	1.000	1.000	1.000	1.000
C9orf106	1	1	1	0	0	0	1.000	1.000	1.000	1.000
CEP250	18	22	15	25	29	13	0.682	0.833	0.448	0.520
CGN	11	11	10	11	12	10	0.909	0.909	0.833	0.909
CPNE1	23	33	19	29	54	21	0.576	0.826	0.389	0.724
CRAT	12	13	12	16	18	14	0.923	1.000	0.778	0.875
CTAG1A	3	3	3	3	3	3	1.000	1.000	1.000	1.000
CTAG1B	3	3	3	3	3	3	1.000	1.000	1.000	1.000
CTAG2	3	3	3	3	3	3	1.000	1.000	1.000	1.000
DDX43	5	5	5	4	4	4	1.000	1.000	1.000	1.000
DEPDC5	37	34	31	44	42	36	0.912	0.838	0.857	0.818
DKC1	17	19	16	19	25	16	0.842	0.941	0.640	0.842
DNASE1L1	10	10	10	14	14	14	1.000	1.000	1.000	1.000
DOLPP1	9	9	9	11	11	11	1.000	1.000	1.000	1.000
DRG1	5	5	5	5	5	5	1.000	1.000	1.000	1.000
EEF1A1	23	25	22	32	36	28	0.880	0.957	0.778	0.875
EIF4ENIF1	17	19	17	20	23	20	0.895	1.000	0.870	1.000
EMD	12	14	11	12	18	8	0.786	0.917	0.444	0.667
ENPP1	8	5	4	9	5	4	0.800	0.500	0.800	0.444

(continue)

Table 1: Details of the first experiment with no errors in the input set.

Gene	G_R vertices	G_F vertices	Correctly predicted vertices	G_R arcs	G_F arcs	Correctly predicted arcs	Sn vertices	PPV vertices	Sn arcs	PPV arcs
ERGIC3	30	33	26	41	46	26	0.788	0.867	0.565	0.634
F10	1	1	1	0	0	0	1.000	1.000	1.000	1.000
F7	10	12	10	12	16	8	0.833	1.000	0.500	0.667
F8	8	8	8	8	8	8	1.000	1.000	1.000	1.000
F8A1	1	1	1	0	0	0	1.000	1.000	1.000	1.000
FAM3A	17	21	15	18	31	14	0.714	0.882	0.452	0.778
FAM50A	12	12	12	13	13	13	1.000	1.000	1.000	1.000
FAM73B	24	25	20	29	35	20	0.800	0.833	0.571	0.690
FAM83C	6	5	5	6	6	6	1.000	0.833	1.000	1.000
FBXO7	13	14	12	17	19	16	0.857	0.923	0.842	0.941
FER1L4	44	42	41	61	59	57	0.976	0.932	0.966	0.934
FLNA	38	40	34	53	57	42	0.850	0.895	0.737	0.792
FOXP4	10	11	10	12	14	12	0.909	1.000	0.857	1.000
FRS3	8	8	8	8	8	8	1.000	1.000	1.000	1.000
FUNDC2	10	10	10	12	11	11	1.000	1.000	1.000	0.917
G6PD	16	19	13	18	24	12	0.684	0.813	0.500	0.667
GAB3	9	10	9	10	12	10	0.900	1.000	0.833	1.000
GDF5	3	3	3	2	2	2	1.000	1.000	1.000	1.000
H2AFB1	1	1	1	0	0	0	1.000	1.000	1.000	1.000
HCFC1	6	7	5	7	9	6	0.714	0.833	0.667	0.857
IER5L	1	1	1	0	0	0	1.000	1.000	1.000	1.000
IKBKG	19	19	17	31	25	15	0.895	0.895	0.600	0.484
IRAK1	23	30	22	29	46	23	0.733	0.957	0.500	0.793
KATNAL1	10	7	6	9	8	6	0.857	0.600	0.750	0.667
KCNQ5	10	10	7	9	11	3	0.700	0.700	0.273	0.333
L1CAM	22	25	21	27	31	22	0.840	0.955	0.710	0.815
LACE1	5	6	4	4	7	3	0.667	0.800	0.429	0.750
LAGE3	1	1	1	0	0	0	1.000	1.000	1.000	1.000
MCF2L	46	48	44	56	58	49	0.917	0.957	0.845	0.875
MDFI	11	11	11	14	14	14	1.000	1.000	1.000	1.000
MECP2	15	16	13	19	21	8	0.813	0.867	0.381	0.421
MMP24	1	1	1	0	0	0	1.000	1.000	1.000	1.000
MOXD1	5	5	5	5	5	5	1.000	1.000	1.000	1.000
MPP1	22	23	22	32	33	32	0.957	1.000	0.970	1.000
MTCP1	6	7	5	8	10	3	0.714	0.833	0.300	0.375
MTO1	14	15	13	19	21	18	0.867	0.929	0.857	0.947
NCR2	5	5	5	6	6	6	1.000	1.000	1.000	1.000
NFS1	17	18	16	22	24	21	0.889	0.941	0.875	0.955
NR2E1	6	6	6	5	5	5	1.000	1.000	1.000	1.000

(continue)

Table 1: Details of the first experiment with no errors in the input set.

Gene	G_R vertices	G_F vertices	Correctly predicted vertices	G_R arcs	G_F arcs	Correctly predicted arcs	Sn vertices	PPV vertices	Sn arcs	PPV arcs
NUP188	19	21	17	20	23	16	0.810	0.895	0.696	0.800
OPN1LW	3	3	3	3	3	3	1.000	1.000	1.000	1.000
OPN1MW	3	3	3	3	3	3	1.000	1.000	1.000	1.000
OSTM1	11	12	9	12	15	8	0.750	0.818	0.533	0.667
PCDH15	18	19	18	23	27	23	0.947	1.000	0.852	1.000
PGC	7	7	7	8	8	8	1.000	1.000	1.000	1.000
PHYHD1	12	13	11	18	19	14	0.846	0.917	0.737	0.778
PIP5K1A	17	19	15	19	23	13	0.789	0.882	0.565	0.684
PISD	18	23	15	19	32	15	0.652	0.833	0.469	0.789
PLXNA3	14	16	13	15	17	12	0.813	0.929	0.706	0.800
POGZ	22	22	21	27	28	27	0.955	0.955	0.964	1.000
PPP2R4	25	25	25	32	32	32	1.000	1.000	1.000	1.000
PSMB4	11	12	11	15	17	14	0.917	1.000	0.824	0.933
PSMD4	21	23	19	30	31	20	0.826	0.905	0.645	0.667
RBM12	4	5	4	4	7	4	0.800	1.000	0.571	1.000
RBM39	35	41	33	41	58	36	0.805	0.943	0.621	0.878
RENBP	15	16	15	20	23	20	0.938	1.000	0.870	1.000
RFPL2	3	3	3	2	2	2	1.000	1.000	1.000	1.000
RFPL3	1	3	0	0	2	0	0.000	0.000	0.000	1.000
RFPL3S	5	5	5	5	5	5	1.000	1.000	1.000	1.000
RFX5	18	25	14	21	36	12	0.560	0.778	0.333	0.571
RP11-374F3.4	15	15	15	17	17	17	1.000	1.000	1.000	1.000
RPL10	13	16	12	13	25	13	0.750	0.923	0.520	1.000
SEC63	2	2	2	0	0	0	1.000	1.000	1.000	1.000
SELENBP1	21	22	21	26	33	26	0.955	1.000	0.788	1.000
SFI1	42	43	42	54	54	50	0.977	1.000	0.926	0.926
SH3GLB2	16	18	14	21	26	16	0.778	0.875	0.615	0.762
SLC10A3	7	7	7	9	9	9	1.000	1.000	1.000	1.000
SLC5A1	3	5	2	2	4	1	0.400	0.667	0.250	0.500
SLC5A4	1	1	1	0	0	0	1.000	1.000	1.000	1.000
SNX27	5	6	5	5	6	5	0.833	1.000	0.833	1.000
SNX3	5	5	5	7	7	7	1.000	1.000	1.000	1.000
SPAG4	13	14	12	17	19	13	0.857	0.923	0.684	0.765
STAG2	31	26	22	58	37	21	0.846	0.710	0.568	0.362
SYN3	19	19	19	19	19	19	1.000	1.000	1.000	1.000
TAZ	25	30	23	28	48	24	0.767	0.920	0.500	0.857
TFEB	20	24	17	22	29	8	0.708	0.850	0.276	0.364
TIMP3	3	5	3	2	6	1	0.600	1.000	0.167	0.500
TKTL1	9	10	8	10	12	9	0.800	0.889	0.750	0.900

(continue)

Table 1: Details of the first experiment with no errors in the input set.

Gene	G_R vertices	G_F vertices	Correctly predicted vertices	G_R arcs	G_F arcs	Correctly predicted arcs	Sn vertices	PPV vertices	Sn arcs	PPV arcs
TUFT1	12	12	12	14	14	14	1.000	1.000	1.000	1.000
USP49	6	6	6	7	7	7	1.000	1.000	1.000	1.000
VPS72	4	6	3	3	7	2	0.500	0.750	0.286	0.667
YWHAH	6	6	6	7	7	7	1.000	1.000	1.000	1.000
ZNF687	10	10	10	13	13	13	1.000	1.000	1.000	1.000
Average							0.880	0.927	0.786	0.868

Table 2: Details of the second experiment with $c = 4$ and 16% of errors in the input set. Columns "G_R vertices" and "G_F vertices" are the number of vertices in the graph G_R and G_F, respectively. Column "Correctly predicted vertices" represents the number of vertices of G_F that are correctly predicted by G_R. The next 3 columns ("G_R arcs", "G_F arcs" and "Correctly predicted arcs") represent the number of arcs in the two graphs and number of arcs of G_F that are correctly predicted by G_R. The last 4 columns are the values of Sn and PPV on vertices and arcs.

Gene	G_R vertices	G_F vertices	Correctly predicted vertices	G_R arcs	G_F arcs	Correctly predicted arcs	Sn vertices	PPV vertices	Sn arcs	PPV arcs
ARHGAP4	32	34	31	45	47	41	0.912	0.969	0.872	0.911
ATP11A	20	20	16	21	23	8	0.800	0.800	0.348	0.381
ATP6AP1	9	9	7	11	12	8	0.778	0.778	0.667	0.727
AVPR2	7	7	7	8	8	8	1.000	1.000	1.000	1.000
BPIL2	5	5	5	5	5	5	1.000	1.000	1.000	1.000
BRCC3	8	11	8	9	15	9	0.727	1.000	0.600	1.000
C20orf173	5	5	5	6	6	6	1.000	1.000	1.000	1.000
C22orf24	5	5	5	5	5	5	1.000	1.000	1.000	1.000
C22orf28	11	11	11	12	13	12	1.000	1.000	0.923	1.000
C22orf30	12	12	10	15	15	11	0.833	0.833	0.733	0.733
C6orf150	7	7	7	7	7	7	1.000	1.000	1.000	1.000
C9orf106	1	1	1	0	0	0	1.000	1.000	1.000	1.000
CEP250	18	22	14	25	29	13	0.636	0.778	0.448	0.520
CGN	12	11	9	12	12	8	0.818	0.750	0.667	0.667
CPNE1	24	33	18	33	54	18	0.545	0.750	0.333	0.545
CRAT	12	13	12	16	18	14	0.923	1.000	0.778	0.875
CTAG1A	3	3	3	3	3	3	1.000	1.000	1.000	1.000

(continue)

Table 2: Details of the second experiment with $c = 4$ and 16% of errors in the input set.

Gene	G_R vertices	G_F vertices	Correctly predicted vertices	G_R arcs	G_F arcs	Correctly predicted arcs	Sn vertices	PPV vertices	Sn arcs	PPV arcs
CTAG1B	3	3	3	3	3	3	1.000	1.000	1.000	1.000
CTAG2	3	3	3	3	3	3	1.000	1.000	1.000	1.000
DDX43	5	5	5	4	4	4	1.000	1.000	1.000	1.000
DEPDC5	38	34	31	44	42	36	0.912	0.816	0.857	0.818
DKC1	17	19	15	19	25	14	0.789	0.882	0.560	0.737
DNASE1L1	10	10	10	14	14	14	1.000	1.000	1.000	1.000
DOLPP1	9	9	9	11	11	11	1.000	1.000	1.000	1.000
DRG1	5	5	5	5	5	5	1.000	1.000	1.000	1.000
EEF1A1	23	25	22	32	36	28	0.880	0.957	0.778	0.875
EIF4ENIF1	17	19	17	20	23	20	0.895	1.000	0.870	1.000
EMD	14	14	12	14	18	10	0.857	0.857	0.556	0.714
ENPP1	8	5	4	9	5	4	0.800	0.500	0.800	0.444
ERGIC3	33	33	23	49	46	16	0.697	0.697	0.348	0.327
F10	1	1	1	0	0	0	1.000	1.000	1.000	1.000
F7	12	12	10	18	16	8	0.833	0.833	0.500	0.444
F8	8	8	8	8	8	8	1.000	1.000	1.000	1.000
F8A1	1	1	1	0	0	0	1.000	1.000	1.000	1.000
FAM3A	17	21	15	17	31	13	0.714	0.882	0.419	0.765
FAM50A	12	12	12	13	13	13	1.000	1.000	1.000	1.000
FAM73B	26	25	19	30	35	19	0.760	0.731	0.543	0.633
FAM83C	6	5	5	7	6	6	1.000	0.833	1.000	0.857
FBXO7	13	14	12	17	19	16	0.857	0.923	0.842	0.941
FER1L4	44	42	40	61	59	53	0.952	0.909	0.898	0.869
FLNA	38	40	32	53	57	36	0.800	0.842	0.632	0.679
FOXP4	10	11	9	12	14	10	0.818	0.900	0.714	0.833
FRS3	9	8	8	9	8	8	1.000	0.889	1.000	0.889
FUNDC2	10	10	10	12	11	11	1.000	1.000	1.000	0.917
G6PD	16	19	13	18	24	12	0.684	0.813	0.500	0.667
GAB3	9	10	9	10	12	10	0.900	1.000	0.833	1.000
GDF5	3	3	3	2	2	2	1.000	1.000	1.000	1.000
H2AFB1	1	1	1	0	0	0	1.000	1.000	1.000	1.000
HCFC1	6	7	5	7	9	6	0.714	0.833	0.667	0.857
IER5L	1	1	1	0	0	0	1.000	1.000	1.000	1.000
IKBKG	21	19	18	28	25	17	0.947	0.857	0.680	0.607
IRAK1	23	30	20	29	46	18	0.667	0.870	0.391	0.621
KATNAL1	10	7	6	12	8	6	0.857	0.600	0.750	0.500
KCNQ5	12	10	7	13	11	4	0.700	0.583	0.364	0.308
L1CAM	22	25	21	27	31	22	0.840	0.955	0.710	0.815
LACE1	5	6	4	4	7	3	0.667	0.800	0.429	0.750

(continue)

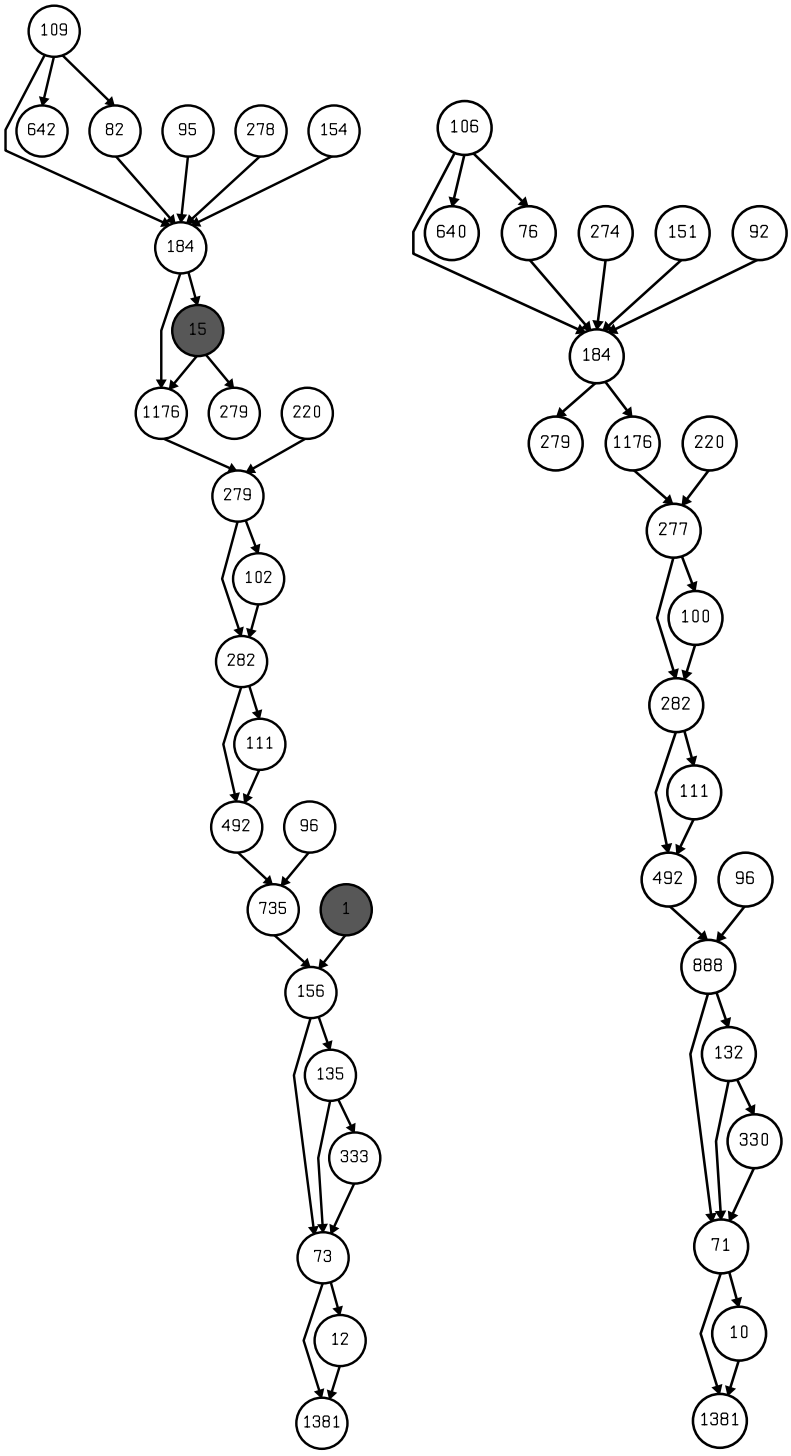
Table 2: Details of the second experiment with $c = 4$ and 16% of errors in the input set.

Gene	G_R vertices	G_F vertices	Correctly predicted vertices	G_R arcs	G_F arcs	Correctly predicted arcs	Sn vertices	PPV vertices	Sn arcs	PPV arcs
LAGE3	1	1	1	0	0	0	1.000	1.000	1.000	1.000
MCF2L	47	48	44	56	58	46	0.917	0.936	0.793	0.821
MDFI	11	11	11	14	14	14	1.000	1.000	1.000	1.000
MECP2	15	16	13	19	21	8	0.813	0.867	0.381	0.421
MMP24	1	1	1	0	0	0	1.000	1.000	1.000	1.000
MOXD1	5	5	5	5	5	5	1.000	1.000	1.000	1.000
MPP1	22	23	22	32	33	32	0.957	1.000	0.970	1.000
MTCP1	6	7	5	8	10	3	0.714	0.833	0.300	0.375
MTO1	15	15	12	20	21	13	0.800	0.800	0.619	0.650
NCR2	5	5	5	6	6	6	1.000	1.000	1.000	1.000
NFS1	18	18	16	23	24	21	0.889	0.889	0.875	0.913
NR2E1	6	6	6	5	5	5	1.000	1.000	1.000	1.000
NUP188	19	21	17	20	23	16	0.810	0.895	0.696	0.800
OPN1LW	3	3	3	3	3	3	1.000	1.000	1.000	1.000
OPN1MW	3	3	3	3	3	3	1.000	1.000	1.000	1.000
OSTM1	11	12	9	12	15	8	0.750	0.818	0.533	0.667
PCDH15	19	19	18	24	27	23	0.947	0.947	0.852	0.958
PGC	7	7	7	8	8	8	1.000	1.000	1.000	1.000
PHYHD1	12	13	10	18	19	12	0.769	0.833	0.632	0.667
PIP5K1A	17	19	15	21	23	13	0.789	0.882	0.565	0.619
PISD	18	23	14	19	32	14	0.609	0.778	0.438	0.737
PLXNA3	14	16	12	15	17	9	0.750	0.857	0.529	0.600
POGZ	22	22	21	27	28	27	0.955	0.955	0.964	1.000
PPP2R4	27	25	23	35	32	23	0.920	0.852	0.719	0.657
PSMB4	11	12	11	15	17	14	0.917	1.000	0.824	0.933
PSMD4	21	23	19	30	31	20	0.826	0.905	0.645	0.667
RBM12	4	5	4	4	7	4	0.800	1.000	0.571	1.000
RBM39	36	41	33	46	58	36	0.805	0.917	0.621	0.783
RENBP	16	16	15	21	23	20	0.938	0.938	0.870	0.952
RFPL2	3	3	3	2	2	2	1.000	1.000	1.000	1.000
RFPL3	1	3	0	0	2	0	0.000	0.000	0.000	1.000
RFPL3S	5	5	5	5	5	5	1.000	1.000	1.000	1.000
RFX5	17	25	13	19	36	10	0.520	0.765	0.278	0.526
RP11-374F3.4	15	15	15	17	17	17	1.000	1.000	1.000	1.000
RPL10	13	16	12	13	25	13	0.750	0.923	0.520	1.000
SEC63	2	2	2	0	0	0	1.000	1.000	1.000	1.000
SELENBP1	20	22	18	24	33	16	0.818	0.900	0.485	0.667
SFI1	42	43	42	54	54	50	0.977	1.000	0.926	0.926
SH3GLB2	16	18	15	21	26	19	0.833	0.938	0.731	0.905

(continue)

Table 2: Details of the second experiment with $c = 4$ and 16% of errors in the input set.

Gene	G_R vertices	G_F vertices	Correctly predicted vertices	G_R arcs	G_F arcs	Correctly predicted arcs	Sn vertices	PPV vertices	Sn arcs	PPV arcs
SLC10A3	7	7	7	9	9	9	1.000	1.000	1.000	1.000
SLC5A1	3	5	2	2	4	1	0.400	0.667	0.250	0.500
SLC5A4	1	1	1	0	0	0	1.000	1.000	1.000	1.000
SNX27	5	6	5	5	6	5	0.833	1.000	0.833	1.000
SNX3	5	5	5	7	7	7	1.000	1.000	1.000	1.000
SPAG4	14	14	13	18	19	15	0.929	0.929	0.789	0.833
STAG2	n.a.	26	n.a.	n.a.	37	n.a.	n.a.	n.a.	n.a.	n.a.
SYN3	19	19	17	19	19	16	0.895	0.895	0.842	0.842
TAZ	25	30	22	28	48	19	0.733	0.880	0.396	0.679
TFEB	20	24	15	22	29	8	0.625	0.750	0.276	0.364
TIMP3	3	5	3	2	6	1	0.600	1.000	0.167	0.500
TKTL1	9	10	8	10	12	9	0.800	0.889	0.750	0.900
TUFT1	12	12	12	14	14	14	1.000	1.000	1.000	1.000
USP49	6	6	6	7	7	7	1.000	1.000	1.000	1.000
VPS72	4	6	3	3	7	2	0.500	0.750	0.286	0.667
YWHAH	6	6	6	7	7	7	1.000	1.000	1.000	1.000
ZNF687	10	10	10	13	13	13	1.000	1.000	1.000	1.000
Average							0.868	0.906	0.765	0.836



(a) Isoform Graph G_F^* (b) RNA-Seq Graph G_R

Figure 3: The isoform graph G_F^* of the gene L1CAM (left) and the graph predicted by our implementation on the same gene (right). The differences between the two graphs consist of two nodes (the gray nodes of the figure) of G_F^* that are associated to short blocks. The misprediction of those nodes also causes in G_R an additional arc (that skips the missing vertex of length 15), and merging of two nodes.